Amendments to the Specification:

Please replace the paragraph that begins on page 6, line 1, with the following paragraph:

Figure 3 illustrates a flow diagram of an exemplary retirement routine 300 in accordance with an embodiment of the invention. The ~~steps~~ operations of the retirement routine are taken after a successful execution of an instruction. In ~~step~~block 302, the retirement logic unit 222 sets the executed flag in ROB entry corresponding to the instruction. In ~~step~~block 304, the retirement logic unit 222 determines whether there is a destination real register 217 for the instruction. If there is no destination real register 217 for the instruction, the retirement routine 300 ends. If, on the other hand, thee is a destination real register 217 for the instruction, in ~~step~~block 306 the retirement logic unit 222 determines whether the real register 217 is designated to undergo the retirement routine 300 in accordance with the invention (i.e., whether the register is one listed in the data commitment table 216).

Please replace the paragraph that begins on page 6, line 22, with the following paragraph:

Accordingly, if in ~~step~~block 306 the retirement logic unit 222 determines that the real register be written to is exempt form the new retirement routine 300, then in ~~step~~block 308 the retirement logic unit 222 causes the copying of the resulting data from the alias register to the real register. Otherwise, in ~~step~~block 308, the retirement logic unit 222 causes the setting of the valid data bit in the ROB entry pertaining to that instruction. In ~~step~~block 310, the retirement logic unit 222 reads the committed value location field of the data commitment table 216 corresponding to the real register to determine if the previous register value is in the real register or in an alias register. If the retirement logic unit 222 determines that the previous register value is in an alias register, in ~~step~~block 314 the retirement logic unit 222 causes a deasserting of the valid data bit of the ROB entry pointed to by the data commitment table 216. Then in ~~step~~block 316 the retirement logic unit 222 causes the writing of the ROB entry index of the instant instruction to the ROB entry index field of the data commitment table 216 corresponding to the real register associated with the new data, and modifies the committed data location field to indicate that the register value is in an alias register pointed to by the corresponding ROB entry index field. If, on the other hand, in ~~step~~block 314 the retirement logic unit 222 determines that

the previous register value is in the RRF ~~216~~217, the retirement logic unit 222 just performs the function specified in ~~step~~block 316 as previously discussed.

Please replace the paragraph that begins on page 7, line 4, with the following paragraph:

The new retirement routine 300 saves an alias register -to-real register copying ~~step~~block (relative to the prior art retirement routine) each time the retirement routine 300 performs ~~step~~block 314. This situation occurs when the same real register is written to (actually written to its alias in the ROB) by two or more instructions within the same instruction window (the size of the ROB). This is substantially different than the prior art retirement routine that makes an alias register-to-real register coy each time an instruction retires. Whereas the new retirement routine 300, avoids some of these copies, and in theory, can eliminate essentially 100 percent of the register writes if the code reuses results extensively, e.g. a long series of "inc eax; inc eax; inc eax, ...". Accordingly, the reduction in real register copying ahs the beneficial results of lower power consumption, extended battery life and a less sophisticated cooling system for the processor, among other benefits.

Please replace the paragraph that begins on page 7, line 23, with the following paragraph:

Specifically, the ~~step~~block in 402 the allocator 214 locates the next ROB entry n for a new instruction. In ~~step~~block 404, the allocator 214 read reads the valid data field of the next ROB entry n to determine whether the corresponding alias register contains valid data. If not, the allocator 214 proceeds to ~~step~~block 412 to add the new instruction into the next ROB entry n. If, however, the valid data field indicates that the next ROB entry n has valid data, in ~~step~~block 406 the allocator 214 causes the content in the alias register of the next ROB entry n to be copies in to the corresponding real register 217. In ~~step~~block 408, the allocator 214 deasserts the valid data bit in the next ROB entry n since the new instruction has not been executed, and therefore the next ROB entry n has yet to have valid data. Then in ~~step~~block 410 the allocator 214 modifies the "committed value location" field of the data commitment table 216 to indicate that the register value for the corresponding real register is now in the real register 217. Finally, in ~~step~~block 412 the allocator 214 causes the new instruction to be added into the next ROB entry n.

Please replace the paragraph that begins on page 8, line 1, with the following paragraph:

In the case that there has been a branch misprediction, or other control flow altering event, like an exception, all the non-committed registers younger than the branch in the ROB 215 are invalid. In the prior art processor system, all non-committed register are discarded by setting the renamer tables to point to all the registers last value to RRF. However, according to the new processor system 200, some of the committed data will reside in the ROB 215. According to the processor 200 of the invention, this can be dealt with in two manners. The first option is to copy the committed data in the ROB 215 to the RRF ~~216~~ 217 in the time the pipeline fills up again. The second option is to make the pointers in the renamer to point to the ROB entry that the data commitment table indicates. For example, if an instruction that writes to the EAX register is committed form the ROB entry index 31, the data commitment table entry corresponding to the EAX will contain the number 31 in the corresponding ROB entry index field. After a branch misprediction, the renamer will now point to the last value of the EAX to ROB entry 31. An instruction that has a source the register EAX, will gets its source renamed to ROB entry 31, so it will get the correct data.